

Phân loại thư rác sử dụng thuật toán SVM

Nhóm 3

B22DCKH024 - Vũ Công Tuấn Dương
B22DCCN768 - Nguyễn Sơn Tùng
B22DCCN479 - Nguyễn Đức Lâm
B22DCCN347 - Trần Đức Hoàng
B22DCCN348 - Trần Huy Hoàng

Ngày 10 tháng 5 năm 2025

- ① Giới thiệu
- ② Lý thuyết
- ③ Cài đặt
- ④ Triển khai và demo

- 1 Giới thiệu
- 2 Lý thuyết
- 3 Cài đặt
- 4 Triển khai và demo

Dữ liệu được lấy trên Kaggle

- Combined Spam Email CSV of 2007 TREC Public Spam Corpus and Enron-Spam Dataset
- 83446 bản ghi email bằng tiếng Anh được phân loại thành 2 nhãn là Spam và Non-spam trong đó số email spam: 43910 và số email ham:(không spam) là 39538

Mục tiêu

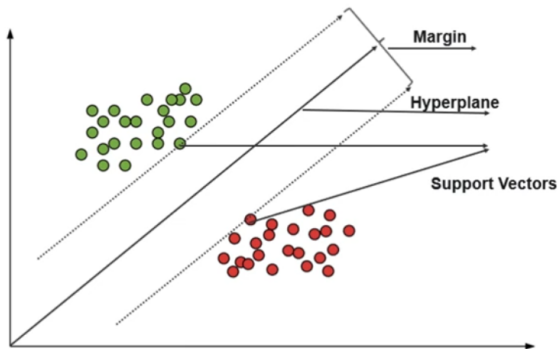
- Xây dựng được mô hình Linear SVM Hard Margin cơ bản để phân loại
- Đánh giá mô hình dựa trên các thang đo như độ chính xác và F1 score
- Demo được trên giao diện web

- 1 Giới thiệu
- 2 Lý thuyết
 - Lý thuyết SVM
 - Thuật toán Pegasos
- 3 Cài đặt
- 4 Triển khai và demo

Hàm mục tiêu của SVM

- Mục tiêu của SVM là tối ưu hóa khoảng cách giữa các điểm dữ liệu của hai lớp.
- Hàm mục tiêu được xây dựng sao cho margin giữa hai lớp là lớn nhất, đồng thời đảm bảo rằng không có điểm nào bị sai phân loại.

Ảnh minh hoạ



Hình 1: Mô tả thuật toán SVM

Hàm mục tiêu của SVM

- Nếu dữ liệu huấn luyện có thể phân tách tuyến tính, có thể chọn hai siêu phẳng song song để phân tách hai lớp dữ liệu, sao cho khoảng cách giữa chúng là lớn nhất có thể.
- Khu vực được giới hạn bởi hai siêu phẳng này được gọi là "margin" (biên). Siêu phẳng có margin lớn nhất là siêu phẳng nằm ở giữa hai siêu phẳng này.

Hàm mục tiêu của SVM

Với một bộ dữ liệu đã chuẩn hóa hoặc chuẩn hóa, các siêu phẳng này có thể được mô tả bằng các phương trình:



$$\mathbf{w}^T \mathbf{x} + b = 1$$

(mọi điểm trên hoặc phía trên ranh giới này thuộc về một lớp với nhãn 1)



$$\mathbf{w}^T \mathbf{x} + b = -1$$

(mọi điểm trên hoặc phía dưới ranh giới này thuộc về lớp còn lại, với nhãn -1).

Về mặt hình học, khoảng cách giữa 2 siêu phẳng này là $\frac{2}{\|w\|}$ nên ta cần tối thiểu hóa $\|w\|$.

Ràng buộc hàm mục tiêu

Cũng cần ngăn không cho các điểm dữ liệu rơi vào margin, vì vậy phải thêm vào ràng buộc sau: với mỗi i , nếu $y_i = 1$, thì phải có:

$$w^T x_i + b \geq 1$$

hoặc nếu $y_i = -1$, thì phải có:

$$w^T x_i + b \leq -1$$

Ràng buộc này yêu cầu mỗi điểm dữ liệu phải nằm ở phía đúng của margin. Hay có thể viết lại là:

$$y_i(w^T x_i + b) \geq 1, \quad \forall 1 \leq i \leq n$$

Bài toán tối ưu

Tóm lại, ta cần giải bài toán tối ưu:

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

với ràng buộc:

$$y_i(w^T x_i + b) \geq 1 \quad \forall i \in \{1, \dots, n\}$$

Cần tìm w và b

Hàm Lagrange

$$\mathcal{L}(w, b, \lambda) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^N \lambda_i (y_i (w \cdot x_i + b) - 1)$$

$\lambda_i \geq 0$ là các hệ số Lagrange

Điều kiện KKT

$$\lambda_i \geq 0 \quad \forall i = 1, \dots, n$$

$$y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) - 1 \geq 0 \quad \forall i = 1, \dots, n$$

$$\lambda_i [y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) - 1] = 0 \quad \forall i = 1, \dots, n$$

Phương trình Lagrange và điều kiện KKT

- $\forall i: \lambda_i = 0$ hoặc $y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) = 1$.
- Các điểm sao cho $\lambda_i > 0$ nằm trên margin được gọi là các vector hỗ trợ (support vectors)

Điều kiện KKT

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^n \lambda_i y^{(i)} \mathbf{x}^{(i)} = \mathbf{0} \quad (1)$$

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{i=1}^n \lambda_i y^{(i)} = 0 \quad (2)$$

Từ (1):

$$\mathbf{w} = \sum_{i=1}^n \lambda_i y^{(i)} \mathbf{x}^{(i)} \quad (3)$$

Từ (2):

$$\sum_{i=1}^n \lambda_i y^{(i)} = 0 \quad (4)$$

Điều kiện KKT

$$\begin{aligned} L(w, b, \lambda_1, \dots, \lambda_n) &\equiv \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \lambda_i \{y^{(i)}(w^\top x^{(i)} + b) - 1\} \\ &= \frac{1}{2} \|w\|^2 - \underbrace{\sum_{i=1}^n \lambda_i y^{(i)} x^{(i)\top}}_{=w^\top} w - \underbrace{\sum_{i=1}^n \lambda_i y^{(i)} b}_{=0} + \sum_{i=1}^n \lambda_i \end{aligned}$$

Điều kiện KKT

$$\begin{aligned} L(w, b, \lambda_1, \dots, \lambda_n) &\equiv \frac{1}{2} \|w\|^2 - \|w\|^2 + \sum_{i=1}^n \lambda_i \\ &= \sum_{i=1}^n \lambda_i - \frac{1}{2} \|w\|^2 \\ &= \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y^{(i)} y^{(j)} x^{(i)\top} x^{(j)} \\ &\equiv \tilde{L}(\lambda_1, \dots, \lambda_n) \end{aligned}$$

Quy về bài toán

Có thể quy về bài toán tìm:

$$\begin{aligned}\hat{\lambda} &= \arg \max_{\lambda} \tilde{L}(\lambda) \\ &= \arg \max_{\lambda} \left\{ \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y^{(i)} y^{(j)} \mathbf{x}^{(i)T} \mathbf{x}^{(j)} \right\}\end{aligned}$$

với điều kiện $\lambda_i \geq 0, \sum_{i=1}^n \lambda_i y^{(i)} = 0, (i = 1, 2, \dots, n)$.

Ước lượng λ bằng phương pháp Gradient Descent

Nguyên lý phương pháp

- Phương pháp Gradient Descent được sử dụng để tìm nghiệm tối ưu λ
- Các giá trị tham số ban đầu $\lambda^{[0]}$ được đặt ngẫu nhiên
- Cập nhật theo hướng gradient (vì đây là bài toán tối đa hóa):

$$\lambda^{[t+1]} = \lambda^{[t]} + \eta \tilde{L}(\lambda)$$

- η là tốc độ học (learning rate)

Biểu diễn vector cho bài toán SVM

Ma trận dữ liệu và vector

$$\mathbf{X}_{[n \times p]} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_p^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_p^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \cdots & x_p^{(n)} \end{pmatrix} = \begin{pmatrix} -\mathbf{x}^{(1)T} - \\ -\mathbf{x}^{(2)T} - \\ \vdots \\ -\mathbf{x}^{(n)T} - \end{pmatrix}$$
$$\mathbf{y}_{[n \times 1]} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix}, \quad \lambda_{[n \times 1]} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix}$$

Ma trận H và phép nhân Hadamard

Định nghĩa ma trận H

$$\mathbf{H}_{[n \times n]} \equiv \mathbf{y}_{[n \times 1]} \mathbf{y}_{[1 \times n]}^T \odot \mathbf{X}_{[n \times p]} \mathbf{X}_{[p \times n]}^T$$

- \odot là tích Hadamard (phép nhân từng phần tử)
- Các phần tử của ma trận: $(H)_{ij} = y^{(i)} y^{(j)} \mathbf{x}^{(i)T} \mathbf{x}^{(j)}$
- H là ma trận đối xứng

Viết lại hàm Lagrangian

Biểu diễn hàm Lagrangian dưới dạng vector

$$\begin{aligned}\tilde{L}(\lambda) &\equiv \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y^{(i)} y^{(j)} \mathbf{x}^{(i)T} \mathbf{x}^{(j)} \\ &= \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j (H)_{ij} \\ &= \|\lambda\| - \frac{1}{2} \lambda^T H \lambda\end{aligned}$$

Với $\|\lambda\| = \sum_{i=1}^n \lambda_i$ là tổng các phần tử của vector λ

Tính toán vector gradient

Đạo hàm của hàm Lagrangian theo λ

$$\begin{aligned}\tilde{L}(\lambda)\lambda &= \lambda\|\lambda\| - \frac{1}{2}\lambda\lambda^T H\lambda \\ &= \mathbf{1} - H\lambda\end{aligned}$$

Trong đó $\mathbf{1}$ là vector cột với tất cả các thành phần bằng 1: $\mathbf{1} = (1, 1, \dots, 1)^T$

Quy tắc cập nhật trong Gradient Descent

Quy tắc cập nhật cho nhân tử λ

$$\lambda^{[t+1]} = \lambda^{[t]} + \eta(\mathbf{1} - H\lambda^{[t]})$$

- Cập nhật này được lặp đi lặp lại cho đến khi hội tụ
- Cần đảm bảo ràng buộc $\lambda_i \geq 0$ bằng cách cắt giá trị âm về 0

Tìm $\hat{\mathbf{w}}$ từ $\hat{\lambda}$

Tính vector trọng số $\hat{\mathbf{w}}$

Từ điều kiện KKT, ta có:

$$\hat{\mathbf{w}} = \sum_{i=1}^n \hat{\lambda}_i y^{(i)} \mathbf{x}^{(i)}$$

- Từ điều kiện KKT (3):

$$\hat{\lambda}_i = 0, \text{ hoặc } y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b) - 1 = 0$$

- Dữ liệu $\mathbf{x}^{(i)}$ được phân loại:

$$\begin{cases} \hat{\lambda}_i \neq 0 \Leftrightarrow \mathbf{x}^{(i)} \text{ là vector hỗ trợ,} \\ \hat{\lambda}_i = 0 \Leftrightarrow \mathbf{x}^{(i)} \text{ không phải là vector hỗ trợ.} \end{cases}$$

Tìm $\hat{\mathbf{w}}$ từ $\hat{\lambda}$

Tính vector trọng số $\hat{\mathbf{w}}$

Từ điều kiện KKT, ta có:

$$\hat{\mathbf{w}} = \sum_{i=1}^n \hat{\lambda}_i y^{(i)} \mathbf{x}^{(i)}$$

- Chỉ tính tổng trên các vector hỗ trợ:

$$\hat{\mathbf{w}} = \sum_{\mathbf{x}^{(i)} \in S} \hat{\lambda}_i y^{(i)} \mathbf{x}^{(i)}$$

Tìm \hat{b} từ các vector hỗ trợ

Tính tham số \hat{b}

$$\begin{aligned}\hat{b} &= \frac{1}{y^{(i)}} - \hat{\mathbf{w}}^T \mathbf{x}^{(i)} \\ &= y^{(i)} - \hat{\mathbf{w}}^T \mathbf{x}^{(i)} \quad (\text{vì } y^{(i)} = 1 \text{ hoặc } -1)\end{aligned}$$

- Trong thực tế, để giảm sai số, tính trung bình trên tất cả các vector hỗ trợ:

$$\hat{b} = \frac{1}{|S|} \sum_{\mathbf{x}^{(i)} \in S} (y^{(i)} - \hat{\mathbf{w}}^T \mathbf{x}^{(i)})$$

- $|S|$ là số lượng vector hỗ trợ

Tóm tắt thuật toán Gradient Descent cho SVM

- ➊ **Khởi tạo:** $\lambda^{[0]}$ ngẫu nhiên, tính ma trận H
- ➋ **Lặp:** Cập nhật λ theo công thức:

$$\lambda^{[t+1]} = \lambda^{[t]} + \eta(\mathbf{1} - H\lambda^{[t]})$$

- ➌ **Áp dụng ràng buộc:** $\lambda_i \geq 0$
- ➍ **Tìm vector hỗ trợ:** $S = \{i : \lambda_i > 0\}$
- ➎ **Tính tham số mô hình:**

$$\hat{\mathbf{w}} = \sum_{\mathbf{x}^{(i)} \in S} \hat{\lambda}_i y^{(i)} \mathbf{x}^{(i)}$$
$$\hat{b} = \frac{1}{|S|} \sum_{\mathbf{x}^{(i)} \in S} (y^{(i)} - \hat{\mathbf{w}}^T \mathbf{x}^{(i)})$$

1 Giới thiệu

② Lý thuyết

Lý thuyết SVM

Thuật toán Pegasus

Thuật toán Pegasus cơ bản

Bước chiếu

Hàm mục tiêu và hàm mất mát Hinge

③ Cài đặt

④ Triển khai và demo

Mô tả bằng mã giả

Algorithm 1 The Pegasos algorithm.

Inputs: a list of example feature vectors X
a list of outputs Y
regularization parameter λ
the number of steps T

$w = (0, \dots, 0)$
for t in $[1, \dots, T]$
 select a position i randomly
 $\eta = \frac{1}{\lambda \cdot t}$
 score = $y_i \cdot (w \cdot x_i)$
 if score < 1
 $w = (1 - \eta \cdot \lambda) \cdot w + (\eta \cdot y_i) \cdot x_i$
 else
 $w = (1 - \eta \cdot \lambda) \cdot w$
the end result is w

Hình 2: Mô tả thuật toán Pegasos cơ bản bằng mã giả

Bước Chiều

Giới hạn tập hợp các nghiệm khả thi trong phạm vi $\frac{1}{\sqrt{\lambda}}$. Để thực hiện điều này, cập nhật \mathbf{w}_t sau mỗi vòng lặp:

$$\mathbf{w}_{t+1} \leftarrow \min \left(1, \frac{1}{\sqrt{\lambda \|\mathbf{w}_{t+1}\|}} \right) \mathbf{w}_{t+1}.$$

Hàm mất mát Hinge

Hàm mất mát hinge có thể được viết rõ ràng hơn như sau:

$$\text{Loss}(\mathbf{w}, \mathbf{x}_i, y_i) = \begin{cases} 1 - y_i \cdot (\mathbf{w} \cdot \mathbf{x}_i) & \text{nếu } y_i \cdot (\mathbf{w} \cdot \mathbf{x}_i) < 1 \\ 0 & \text{khác} \end{cases}$$

Điều mà Pegasus thực hiện là áp dụng một thuật toán tối ưu hóa để tìm \mathbf{w} tối thiểu hóa hàm mục tiêu f :

$$f(\mathbf{w}; A_t) = \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{k} \sum_{i \in A_t} \ell(\mathbf{w}; (\mathbf{x}_i, y_i)).$$

với k là số lượng tập mini-batch.

1 Giới thiệu

② Lý thuyết

③ Cài đặt

Tiền xử lý và chuẩn bị dữ liệu

Load dữ liệu

Tiền xử lý dữ liệu

Chuẩn bị dữ liệu

Mô hình HardMarginSVM

Mô hình SVM tối ưu bởi thuật toán Pegasos

Chạy với bộ dữ liệu ban đầu và lưu mô hình

④ Triển khai và demo

Import thư viện

In [1]:

```
import pandas as pd
import nltk
nltk.download('stopwords')
nltk.download('punkt_tab')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re
from nltk.stem import SnowballStemmer

#Visualization
import matplotlib.pyplot as plt

#Feature Engineering
import string
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

#Evaluation Metric
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, precision_score, recall_score, classification_report
import seaborn as sns
from scipy.sparse import csr_matrix
```

Load dữ liệu

Load dữ liệu

```
In [2]: df = pd.read_csv("https://media.githubusercontent.com/media/PTIT-Projects/ttcs-svm-spam-email/refs/heads/main/dataset/combined_data.csv")
# df = pd.read_csv("https://media.githubusercontent.com/media/PTIT-Projects/ttcs-svm-spam-email/refs/heads/main/dataset/sampled_dataset1000.csv")
```

Import thư viện

```
In [3]: df.head()
```

Out[3]:

	label	text
0	1	ounce feather bowl hummingbird opec moment ala...
1	1	wulvob get your medircations online qnb ikud v...
2	0	computer connection from cnn com wednesday es...
3	1	university degree obtain a prosperous future m...
4	0	thanks for all your answers guys i know i shou...

Xoá những ký tự đặc biệt và số

In [10]:

```
def remove_special_characters(word):  
    return re.sub(r'[^a-zA-Z\s]', '', word)
```

In [11]:

```
text = 'Hello everyone ! I am happy to meet 3 of us, my email is admin@gmail.com'
print(remove_special_characters(text))
```

Hello everyone I am happy to meet of us my email is admin@gmail.com

Xoá các link website trong câu

Xoá các link website trong câu

In [15]:

```
def remove_url(word):
    return re.sub(r"http\S+", "", word)
```

In [16]:

```
text = 'My websites are https://google.com and https://reddit.com'
print(remove_url(text))
```

My websites are and

Áp dụng word_tokenize vào data

In [17]:

```
df['text'] = df['text'].apply(remove_special_characters)
df['text'] = df['text'].apply(remove_url)
df['text'] = df['text'].apply(word_tokenize)
df['text'] = df['text'].apply(remove_stop_words)
df['text'] = df['text'].apply(' '.join)
```

In [18]:

```
df.head()
```

Out[18]:

	label	text
0	1	ounce feather bowl hummingbird opec moment ala...
1	1	wulvob get medircations online qnb ikud viagra...
2	0	computer connection cnn com wednesday escapenu...
3	1	university degree obtain prosperous future mon...
4	0	thanks answers guys know checked rsync manual ...

Snowball Stemmer

Word	Stem
cared	care
university	univers
fairly	fair
easily	easili
singing	sing
sings	sing
sung	sung
singer	singer
sportingly	sport

Snowball Stemmer

```
In [19]: stemmer = SnowballStemmer('english')

def stem_text(text):
    tokens = nltk.word_tokenize(text)

    stemmed_tokens = [stemmer.stem(token) for token in tokens]

    return ' '.join(stemmed_tokens)
```

```
In [20]: df['text'] = df['text'].apply(stem_text)
df.head()
```

Out[20]:

	label	text
0	1	ounc feather bowl hummingbird opec moment alab...
1	1	wulvob get medirc onlin qnb ikud viagra escape...
2	0	comput connect cnn com wednesday escapenumb ma...
3	1	univers degre obtain prosper futur money earn ...
4	0	thank answer guy know check rsync manual would...

TF-IDF vectorization

TF-IDF vectorization

<https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/>

```
[21]: vectorizer = TfidfVectorizer()

X = vectorizer.fit_transform(df['text'])
print(X.shape)
```

(83448, 234926)

Chuẩn bị dữ liệu train và test

Chuẩn bị dữ liệu train và test

In [22]:

```
y = df['label']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```


HardMarginSVM

```

class HardMarginSVM:
    def __init__(self, eta=0.001, epoch=1000, random_state=42):
        self.eta = eta
        self.epoch = epoch
        self.random_state = random_state
        self.is_trained = False
        self.support_vectors = None

    def fit(self, X, y):
        if hasattr(X, "toarray"):
            X = X.toarray()
        self.num_samples = X.shape[0]
        self.num_features = X.shape[1]
        y_unique = np.unique(y)
        if len(y_unique) != 2:
            raise ValueError("Binary classification requires exactly 2 classes")
        if set(y_unique) == {0, 1}:
            y = np.where(y == 0, -1, 1)
        self.w = np.zeros(self.num_features)
        self.b = 0
        rgen = np.random.RandomState(self.random_state)
        self.alpha = rgen.uniform(low=0.0, high=0.01, size=self.num_samples)
        for i in range(self.epoch):
            self._cycle(X, y)
            sv_indices = np.where(self.alpha != 0)[0]
            self.support_vectors = sv_indices
            self.w = np.zeros(self.num_features)
            for i in sv_indices:
                self.w += self.alpha[i] * y[i] * X[i]
            bias_sum = 0
            for i in sv_indices:
                bias_sum += y[i] - np.dot(self.w, X[i])
            self.b = bias_sum / len(sv_indices)
            self.is_trained = True
        return self

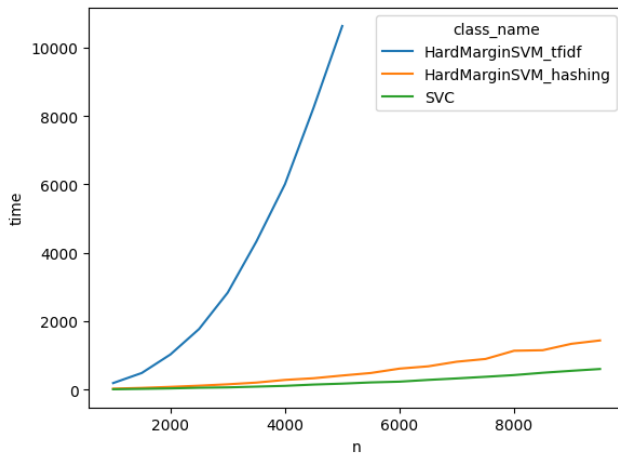
```

Kết quả

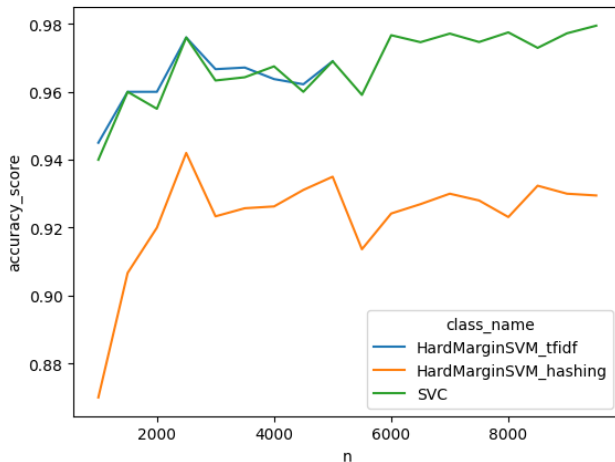
Đang huấn luyện Hard Margin SVM...

	class_name	n	time	accuracy_score	f1_score
0	HardMarginSVM	2000	1011.025832	0.945	0.946602

So sánh thời gian chạy với độ lớn bộ dữ liệu



So sánh độ lớn bộ dữ liệu với độ chính xác



1 Giới thiệu

② Lý thuyết

③ Cài đặt

Tiền xử lý và chuẩn bị dữ liệu

Mô hình HardMarginSVM

Mô hình SVM tối ưu bởi thuật toán Pegasos

Viết mô hình

Chạy thử trên tập dữ liệu

So sánh với HardMarginSVM

Chạy với bộ dữ liệu ban đầu và lưu mô hình

④ Triển khai và demo

SVM với thuật toán Pegasos

```

self.num_samples, self.num_features = X.shape
y_unique = np.unique(y)
if len(y_unique) != 2:
    raise ValueError("Phân loại nhị phân cần 2 nhãn")
if set(y_unique) == {0, 1}:
    y = np.where(y == 0, -1, 1)
self.w = np.zeros(self.num_features, dtype=np.float32)
self.b = 0.0
np.random.seed(self.random_state)
t = 0
previous_objective = float("inf")
for ep in range(1, self.epoch + 1):
    indices = np.random.permutation(self.num_samples)
    for start in range(0, self.num_samples, self.batch_size):
        t += 1
        end = start + self.batch_size
        batch_idx = indices[start:end]
        X_batch = X[batch_idx]
        y_batch = y[batch_idx]

        eta = 1.0 / (self.lambda_param * t)
        margins = y_batch * (X_batch.dot(self.w) + self.b)
        mask = margins < 1
        self.w *= (1 - eta * self.lambda_param)
        if np.any(mask):
            X_violate = X_batch[mask]
            y_violate = y_batch[mask]
            self.w += (eta / self.batch_size) * np.dot(y_violate, X_violate.toarray()) if hasattr(X_violate, "toarray") else X_violate
            self.b += (eta / self.batch_size) * np.sum(y_violate)
        norm_w = np.linalg.norm(self.w)
        factor = min(1, (1.0 / np.sqrt(self.lambda_param))) / (norm_w))
        self.w *= factor

    decision = X.dot(self.w) + self.b
    hinge_losses = np.maximum(0, 1 - y * decision)
    objective = 0.5 * self.lambda_param * np.dot(self.w, self.w) + np.mean(hinge_losses)
    if ep % 10 == 0:
        print(f"Epoch {ep}, Giá trị hàm mục tiêu: {objective:.4f}")
    if abs(previous_objective - objective) < self.tol:
        print(f"Dừng sớm tại epoch {ep}, giá trị hàm mục tiêu thay đổi: {abs(previous_objective - objective):.6f}")
        break
    previous_objective = objective

```

Kết quả

=== Đánh giá SVM tối ưu bằng thuật toán Pegasos ===

==== Kết quả đánh giá SVM tối ưu bằng thuật toán Pegasos ====

Accuracy: 0.9550

Precision: 0.9559

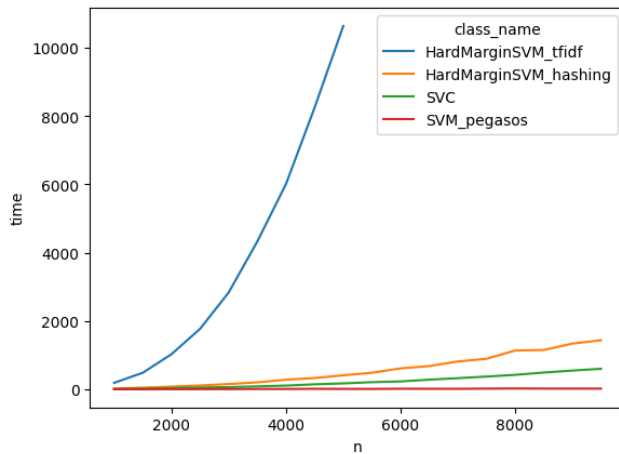
Recall: 0.9559

F1-score: 0.9559

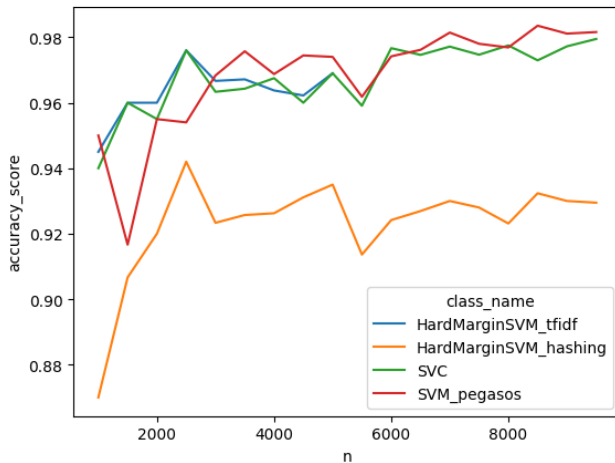
Báo cáo chi tiết:

	precision	recall	f1-score	support
0	0.95	0.95	0.95	196
1	0.96	0.96	0.96	204
accuracy			0.95	400
macro avg	0.95	0.95	0.95	400
weighted avg	0.95	0.95	0.95	400

So sánh thời gian chạy với độ lớn bộ dữ liệu



So sánh độ lớn bộ dữ liệu với độ chính xác



Kết quả

Start preprocessing data

(66758, 206343)

End preprocessing data

Epoch 10, Giá trị hàm mục tiêu: 0.1199

Epoch 20, Giá trị hàm mục tiêu: 0.1176

Dừng sớm tại epoch 22, giá trị hàm mục tiêu thay đổi: 0.000032

	class_name	n	time	prep_time	accuracy_score	f1_score
0	SVM	83448	261.033806	283.776974	0.981186	0.982282

Lưu mô hình và vectorizer

```
model_filename = f'linear_svm.pkl'
vectorizer_filename = f'vectorizer.pkl'

with open(model_filename, 'wb') as model_file:
    pickle.dump(svm_base, model_file)

with open(vectorizer_filename, 'wb') as vectorizer_file:
    pickle.dump(vectorizer, vectorizer_file)
```


Mô tả cách triển khai

- Sử dụng streamlit để tạo giao diện trên web và xử lý phần mô hình
- Load 2 file nhị phân model và vectorizer để tiến hành dự đoán đầu vào
- Sử dụng docker để đóng gói

Demo phân loại email tiếng Anh spam

Nhập nội dung email:

Add Information, Not Clutter with Shape Data

Need to add more information to your diagram, but don't want to end up with gigantic shapes with huge blocks of text? Shape data lets you enrich a shape with information and it's viewable as a tooltip. You can add cost, description, asset number, manufacturer, and so much more. Add information like asset numbers and descriptions without clutter

Kiểm tra

 Email có khả năng là SPAM

Văn bản sau khi được tiền xử lý:

add inform not clutter shape data need add inform diagram dont want end gigant shape huge block text
shape data let enrich shape inform viewabl tooltip you add cost descript asset number manufactur much
add inform like asset number descript without clutter

Demo phân loại email tiếng Anh spam

Nhập nội dung email:

The report suggests that universities will need to look beyond their traditional international student recruitment markets, given that demand in India is slowing and higher education is improving in quality in east Asia. They may also need to consider offering more cost-effective options, the report notes.

Kiểm tra

☒ Email có khả năng không phải là SPAM

Văn bản sau khi được tiền xử lý:

the report suggest univrs need look beyond tradit intern student recruit market given demand india slow
higher educ improv qualiti east asia they may also need consid offer costeffect option report note

Cảm ơn cô đã lắng nghe!